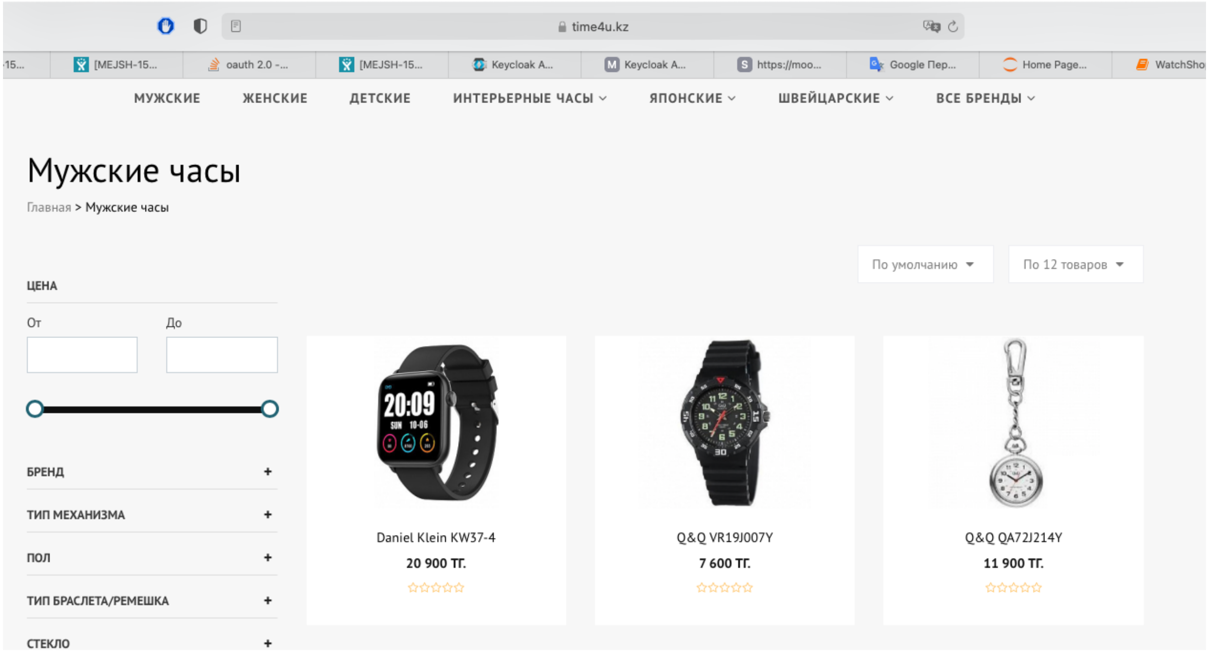
Project
Mohamed Ibrahim

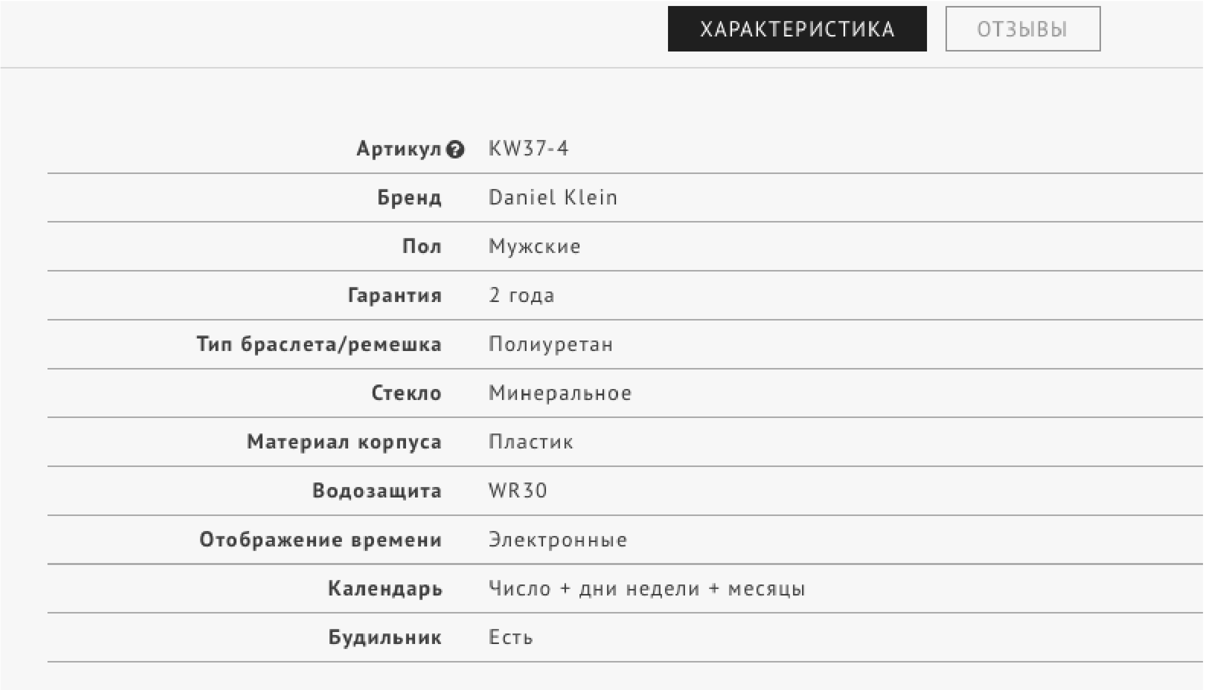
Introduction

This project I have analyzed data from http://watchshop.kz/ and https://time4u.kz. There were problems with the first site that it was not possible to inspect the site to view the tags, which created a lot of problems. As a result, there was a second site where the data met the conditions.

Let's move on to the scraping itself, that is, everything is very simple, using python tools, that is, beautifulSoup you can get data from the site itself.

First of all, I studied the site, it is simple and intuitive, there is a main page, and a page with a set of clocks. I chose a men's watch. Data on men's watches is only 60 pages, each page contains 12 products, in total 720 pieces. The page stores data such as price and brand name, and detailed information is stored in a separate page dedicated to a specific product.



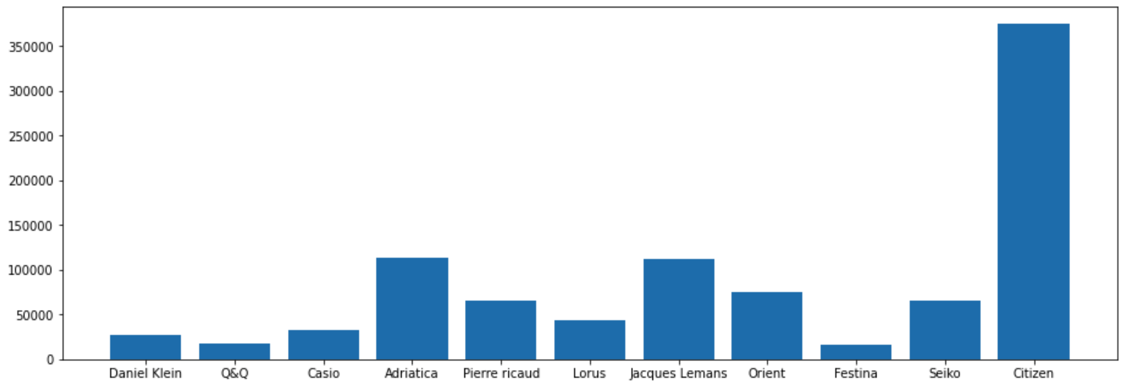


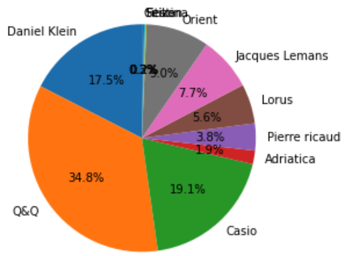
Артикул	KW37-4
Бренд	Daniel Klein
Пол	Мужские
Гарантия	2 года
Тип браслета/ремешка	Полиуретан
Стекло	Минеральное
Материал корпуса	Пластик
Водозащита	WR30
Отображение времени	Электронные
Календарь	Число + дни недели + месяцы
Будильник	Есть

The detailed page contains information such as Brand, Warranty, Bracelet type, Glass, Case material, Water resistance, Calendar, Alarm, Time display. I assumed that the number of such fields should be enough, and proceeded to scrape. First of all, I created a project in Jupiter, loaded all the necessary libraries. And from the main page I collected all the links to the detailed page for each product. After, from each detailed page, I also collected all the characteristics. I wrote the received data to a file in csv format, so that every time I do not scrap the site, as it takes some time. At this point, the data collection phase is over.

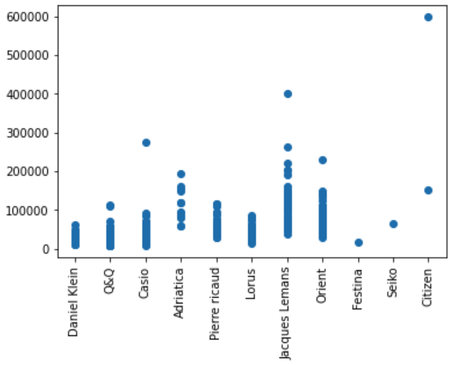
2. Data visualization

First of all, I wanted to see the ratio of prices to brands, which brand is the most expensive and which is the cheapest. It turned out that Citizen is expensive, the average price for a watch is 325,000 tenge, and the cheapest are Q&Q and Festina, about 10,000 tenge. Calculating the average price did not give a complete picture, so I decided to see the relationship between brands in the store. The first three brands of K & K, Casio and Denel Klein are budget brands, which means that most of the proceeds come from the budget segment. I also wanted to emphasize the range of prices by brands, what is the minimum price and the maximum. This is where the picture becomes more obvious. The top 3 expensive are Citizen, Jacques Lemans and Casio. And the most interesting thing is what affects the price of this watch, for this I made a visualization matrix, which shows the correlation between the characteristics. True, this matrix did not really show anything.

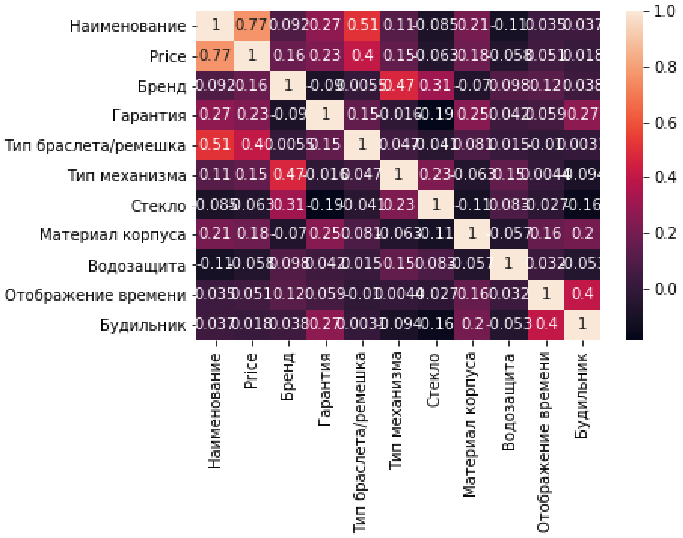




```
plt.scatter(df['Бренд'].values, df['Price'].values)
plt.xticks(rotation=90)
plt.show()
```

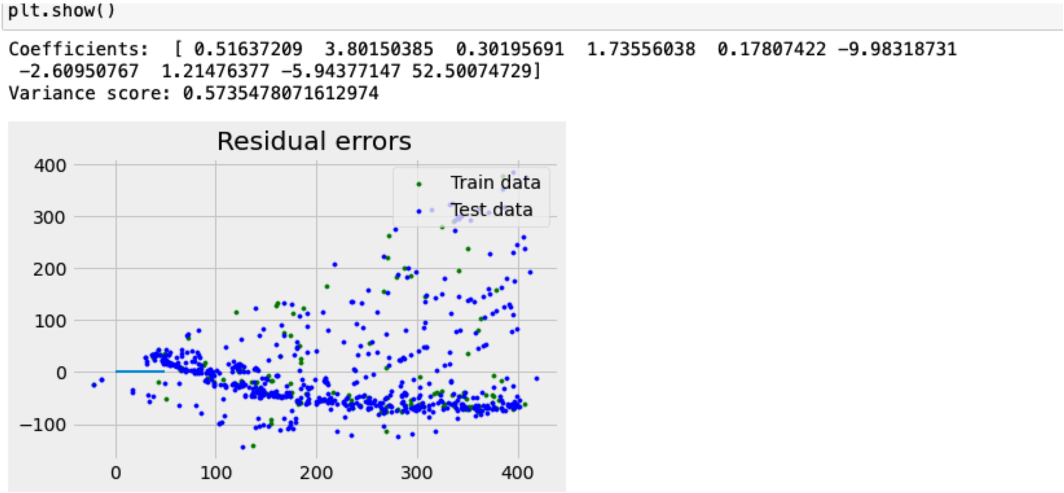


	Price
count	720.000000
mean	39542.858333
std	43388.543267
min	7100.000000
25%	16900.000000
50%	26895.000000
75%	44623.000000
max	600000.000000



3. Data Analytics

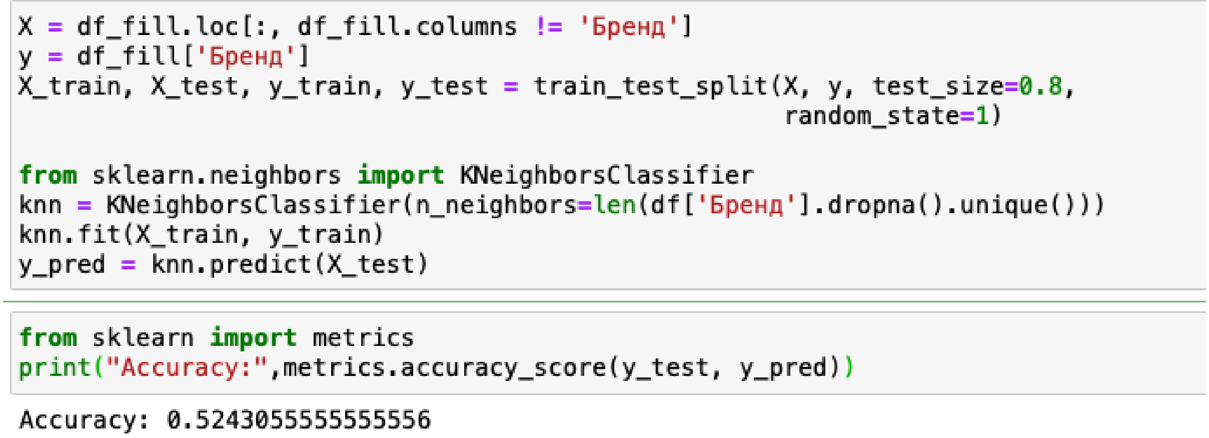
First, I try to analyze data with correlation matrix, to find out related variables. So I tried a model with Price vs other features, to find out accuracy. So I used sklearn libraries to divide into Trains and Test dataset, and show the result in plot.



```
plt.show()
```

```
Coefficients:  [ 0.51637209  3.80150385  0.30195691  1.73556038  0.17807422 -9.98318731
 -2.60950767  1.21476377 -5.94377147 52.50074729]
Variance score: 0.5735478071612974
```

Accuracy is about 57 percent, which is not so good (from correlation matrix we can see that relation between them are poor).



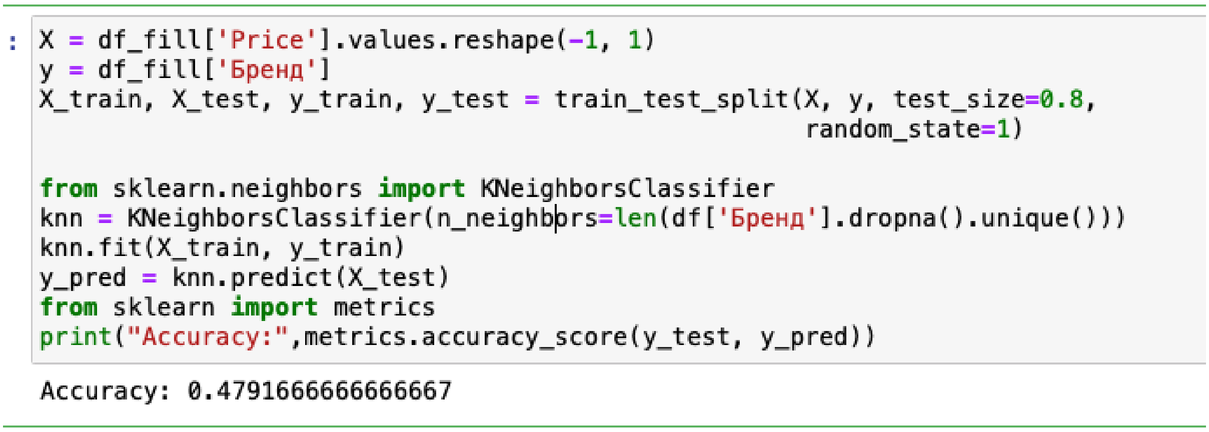
```
X = df_fill.loc[:, df_fill.columns != 'Бренд']
y = df_fill['Бренд']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8,
                                                    random_state=1)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=len(df['Бренд'].dropna().unique()))
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.5243055555555556
```

In my opinion, Classifications, namely CNN, should have been suitable for this task, since goods need to be classified by brands. Here I also divided the data as Brand for Y, and other parameters as X with a ratio of 80 for training, and the remaining 20 for the test. Here the calculations showed 52 percent, just below the Linear Regression. I thought it would be higher. Since many parameters in general were not filled in, I had to fill in with an average value, which spoiled the real picture. And here many parameters intersect with each other, which also complicates the classification. But even so, I think the result is not so bad.



```
: X = df_fill['Price'].values.reshape(-1, 1)
  y = df_fill['Бренд']
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8,
                                                      random_state=1)

  from sklearn.neighbors import KNeighborsClassifier
  knn = KNeighborsClassifier(n_neighbors=len(df['Бренд'].dropna().unique()))
  knn.fit(X_train, y_train)
  y_pred = knn.predict(X_test)
  from sklearn import metrics
  print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

  Accuracy: 0.4791666666666667
```

Then I tried to calculate price vs brand, but accuracy get lower, about 48 percent.

Conclusion.

I tried to get more or less clean data to identify the relationship between the parameters, especially for the Brand and Price parameters. I understand that there are ways for improvement, which I see only now, at the end, namely, it was necessary to make a separate parameter for the Null values, so that the correlation table would give a more realistic picture than now. I think this would help increase the calculations for classifications and linear regressions.

Nevertheless, the obtained data fully shows the picture of goods in the store. We learned that budget models are predominantly dominant and rather cover 80 percent of income.